IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Boucher et al. — Serial No.: Unknown

Filed: August 4, 2003 — Examiner: Unknown

Docket No: ALA-008G — GAU: Unknown

Assignee: Alacritech Inc.

Title: METHOD AND APPARATUS FOR DATA RE-ASSEMBLY WITH A HIGH PERFORMANCE NETWORK INTERFACE

Mail Stop PATENT APPLICATION
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

August 4, 2003

REQUEST TO PROVOKE INTERFERENCE

Sir:

Applicants hereby seek to have an interference declared between the present application and U.S. Patent No. 6,480,489, issued November 12, 2002.

U.S. Patent No. 6,480,489 to Muller et al. issued from U.S. Patent Application Serial No. 09/260,333, filed March 1, 1999.

The present application claims the benefit under 35 U.S.C. §120 of (is a continuation of) U.S. Patent Application Serial No. 10/005,536, filed November 7, 2001, which in turn claims the benefit under 35 U.S.C. §120 of (is a continuation of) U.S. Patent Application Serial No. 09/384,792, filed August 27, 1999, now U.S. Patent No. 6,434,620, which in turn: 1) claims the benefit under 35 U.S.C. §119 of Provisional Patent Application Serial No. 60/098,296, filed August 27, 1998, 2) claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. Patent Application Serial No. 09/067,544, filed April 27, 1998, now U.S. Patent No. 6,226,680, and 3) claims the benefit under 35 U.S.C. §120 of (is a

continuation-in-part of) U.S. Patent Application Serial No. 09/141,713, filed August 28, 1998, now U.S. Patent No. 6,389,479.

U.S. Patent No. 6,226,680 and U.S. Patent No. 6,389,479 both claim the benefit under 35 U.S.C. §119 of Provisional Patent Application Serial No. 60/061,809, filed October 14, 1997. The present application also claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. Patent Application Serial No. 09/464,283, filed December 15, 1999, now U.S. Patent No. 6,427,173, and claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. Patent Application Serial No. 09/514,425, filed February 28, 2000, now U.S. Patent No. 6,427,171.

Claims 1-16, 18-33 and 35-40 of the present application are copied from Claims 1-16, 18-33 and 35-40 of U.S. Patent No. 6,480,489, respectively.

REQUIREMENTS OF 37 CFR 607:

Per rule 607(a)(1), the patent is U.S. Patent No. 6,480,489.

Per rule 607(a)(2), Applicants submit Counts 1-37 as set forth below.

Count #1 (corresponds exactly to Claim 1 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area.

Count #2 (corresponds to Claim 2 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said hybrid storage area is substantially equal in size to one memory page of said host computer.

Count #3 (corresponds to Claim 3 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said predetermined size is approximately 256 bytes.

Count #4 (corresponds to Claim 4 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

receiving a code associated with said first packet, said code indicating that a data portion of said first packet is re-assembleable;

wherein said data storage area is a re-assembly data storage area.

Count #5 (corresponds to Claim 5 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

receiving a code associated with said second packet, said code indicating that a data portion of said second packet is not re-assembleable;

wherein said data storage area is a non-re-assembly data storage area.

Count #6 (corresponds to Claim 6 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

receiving a code with said first packet, said code indicating that said first packet does not contain a data portion.

Count #7 (corresponds to Claim 7 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

receiving a code with said second packet, said code indicating that said second packet is smaller than said predetermined size.

Count #8 (corresponds to Claim 8 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

padding said hybrid storage area to align said header portion with a predetermined index in said hybrid storage area.

Count #9 (corresponds to Claim 9 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow.

Count #10 (corresponds exactly to Claim 10 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area.

Count #11 (corresponds to Claim 11 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area;

wherein said re-assembly storage area is substantially equal in size to one memory page of said host computer.

Count #12 (corresponds to Claim 12 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area;

wherein said re-assembly storage area is used to store only data portions of packets in said communication flow.

Count #13 (corresponds to Claim 13 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area; and

retrieving an identifier of a first storage area in a host computer memory.

Count #14 (corresponds to Claim 14 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area;

retrieving an identifier of a first storage area in a host computer memory; and

placing said first storage area identifier in a data structure configured to hold storage area identifiers;

wherein said first storage area identifier is identifiable by an index in said data structure.

Count #15 (corresponds to Claim 15 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area;

retrieving an identifier of a first storage area in a host computer memory; and

placing said first storage area identifier in a data structure configured to hold storage area identifiers; and

using said index to identify said first storage area for storing a portion of said first packet;

wherein said first storage area identifier is identifiable by an index in said data structure.

Count #16 (corresponds to Claim 16 of the '489 patent):

A method of transferring multiple packets in a communication flow into a host computer, comprising:

receiving a first packet at a network interface for transfer to a host computer;

identifying a communication flow comprising said first packet;

receiving a second packet at said network interface, wherein said communication flow further

comprises said second packet;

associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;

receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;

storing a header portion of said first packet in a header storage area;

storing said data portion of said first packet in a re-assembly storage area; and storing a data portion of said second packet in said re-assembly storage area;

retrieving an identifier of a first storage area in a host computer memory; and

placing said first storage area identifier in a data structure configured to hold storage area identifiers; and

configuring a descriptor to store said index of said first storage area identifier to inform said host computer of the use of said first storage area to store one of said header portion and said data portion;

wherein said first storage area identifier is identifiable by an index in said data structure and said first storage area comprises one of said header storage area and said re-assembly storage area.

Count #17 (corresponds to Claim 18 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

parsing a header portion of said first packet;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow.

Count #18 (corresponds to Claim 19 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

parsing a header portion of said first packet;

wherein said first packet belongs to a first communication flow, said second packet belongs to a second communication flow and said parsing comprises:

retrieving an identifier of a source of said first packet; and

retrieving an identifier of a destination of said first packet.

Count #19 (corresponds to Claim 20 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

parsing a header portion of said first packet;

wherein said first packet belongs to a first communication flow, said second packet belongs to a second communication flow and said parsing comprises:

retrieving an identifier of a source of said first packet; and

retrieving an identifier of a destination of said first packet;

wherein said identifying a communication flow comprises assembling said source identifier and said destination identifier to form a flow key.

Count #20 (corresponds to Claim 21 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

parsing a header portion of said first packet;

wherein said first packet belongs to a first communication flow, said second packet belongs to a second communication flow and said parsing comprises:

retrieving an identifier of a source of said first packet; and

retrieving an identifier of a destination of said first packet;

wherein said identifying a communication flow comprises assembling said source identifier and said destination identifier to form a flow key and obtaining an index of said flow key in a database of flow keys.

Count #21 (corresponds to Claim 22 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

and said identifying comprises:

receiving a virtual connection identifier, wherein a communication flow comprising said first packet can be identified by said virtual connection identifier.

Count #22 (corresponds to Claim 23 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

and said storing a header portion comprises:

retrieving an index of a header storage area of a host computer;

retrieving an identifier of a location in said header storage area; and

storing a header portion of said first packet at said location in said header storage area.

Count #23 (corresponds to Claim 24 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

and said storing a header portion comprises:

retrieving an index of a header storage area of a host computer;

retrieving an identifier of a location in said header storage area; and

storing a header portion of said first packet at said location in said header storage area; and

said header storage area index comprises an index of said header storage area within a collection of storage area identifiers.

Count #24 (corresponds to Claim 25 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

and said storing a header portion comprises:

retrieving an index of a header storage area of a host computer;

retrieving an identifier of a location in said header storage area; and

storing a header portion of said first packet at said location in said header storage area;

wherein said header storage area index comprises an index of said header storage area within a collection of storage area identifiers; and

said identifier of a location in said header storage area comprises an address within said header storage area.

Count #25 (corresponds to Claim 26 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

wherein said storing said data portion comprises:

retrieving an identifier of a re-assembly storage area of a host computer;

retrieving an identifier of a location in said re-assembly storage area; and

storing said data portion of said first packet at said location in said re-assembly storage area.

Count #26 (corresponds to Claim 27 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow;

wherein said storing said data portion comprises:

retrieving an identifier of a re-assembly storage area of a host computer;

retrieving an identifier of a location in said re-assembly storage area; and

storing said data portion of said first packet at said location in said re-assembly storage area;

wherein said retrieving an identifier of a re-assembly storage area comprises reading an entry in a re-assembly storage table, said entry corresponding to said communication flow.

Count #27 (corresponds to Claim 28 of the '489 patent):

A method of storing a portion of a packet in a host computer memory, comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer;

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area; and

parsing a header portion of said first packet;

wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow, and said data portion is of unknown size prior to said parsing.

Count #28 (corresponds exactly to Claim 29 of the '489 patent):

A network interface for transferring a packet from a network to a host computer, comprising:

a parser configured to examine one or more packets received from a network;

a flow manager configured to manage a first communication flow comprising said one or more packets;

a header storage area configured to store header portions of said one or more packets;

a first re-assembly storage area configured to store data portions of said one or more packets;

a second re-assembly storage area; and

a transfer module configured to:

transfer a data portion of a first packet of a second communication flow into said second re-assembly storage area if said first packet is larger than a predetermined size; and

otherwise, transfer said first packet into said header storage area.

Count #29 (corresponds exactly to Claim 30 of the '489 patent):

A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet from a network to a host computer, the method comprising:

receiving a first packet at a communication interface;

receiving a second packet at said communication interface;

storing a header portion of said first packet in a hybrid storage area of a host computer;

if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and

if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area.

Count #30 (corresponds exactly to Claim 31 of the '489 patent):

31. A method of transferring a packet from a communication link to a host computer, comprising:

parsing a first packet received from a communication link to determine if the first packet conforms to a predetermined set of communication protocols;

generating, from identifiers of a source and a destination of the first packet extracted from the first packet, a flow key to identify a communication flow comprising the first packet;

updating a flow database to track a status of the communication flow;

associating a first operation code with the first packet to indicate whether the first packet is re-assembleable with another packet in the communication flow;

maintaining a plurality of storage areas for transferring packets to a host computer, including:

a first storage area configured to store data portions of packets in the communication flow;

a second storage area configured to store a packet having an associated operation code indicating that the packet is not re-assembleable; and

a third storage area configured to store packets less than a predetermined size and headers of packets having data portions stored in said first storage area; and

storing the first packet in one or more of said plurality of storage areas for transfer to the host computer.

Count #31 (corresponds to Claim 32 of the '489 patent):

A method of transferring a packet from a communication link to a host computer, comprising:

parsing a first packet received from a communication link to determine if the first packet conforms to a predetermined set of communication protocols;

generating, from identifiers of a source and a destination of the first packet extracted from the first packet, a flow key to identify a communication flow comprising the first packet;

updating a flow database to track a status of the communication flow;

associating a first operation code with the first packet to indicate whether the first packet is re-assembleable with another packet in the communication flow;

maintaining a plurality of storage areas for transferring packets to a host computer, including:

a first storage area configured to store data portions of packets in the communication flow;

a second storage area configured to store a packet having an associated operation code indicating that the packet is not re-assembleable; and

a third storage area configured to store packets less than a predetermined size and headers of packets having data portions stored in said first storage area; and

storing the first packet in one or more of said plurality of storage areas for transfer to the host computer; wherein said storing comprises:

selecting one or more storage areas of said plurality of storage areas on the basis of said operation code.

Count #32 (corresponds to Claim 33 of the '489 patent):

A method of transferring a packet from a communication link to a host computer, comprising:

parsing a first packet received from a communication link to determine if the first packet conforms to a predetermined set of communication protocols;

generating, from identifiers of a source and a destination of the first packet extracted from the first packet, a flow key to identify a communication flow comprising the first packet;

updating a flow database to track a status of the communication flow;

associating a first operation code with the first packet to indicate whether the first packet is re-assembleable with another packet in the communication flow;

maintaining a plurality of storage areas for transferring packets to a host computer, including:

a first storage area configured to store data portions of packets in the communication flow;

a second storage area configured to store a packet having an associated operation code indicating that the packet is not re-assembleable; and

a third storage area configured to store packets less than a predetermined size and headers of packets having data portions stored in said first storage area; and

storing the first packet in one or more of said plurality of storage areas for transfer to the host computer;

wherein each of said plurality of storage areas is substantially equal in size to one memory page of the host computer.

Count #33 (corresponds to Claim 35 of the '489 patent):

A network interface for transferring a packet from a network to a host computer, comprising:

a parser configured to examine one or more packets received from a network;

a flow manager configured to manage a first communication flow comprising said one or more packets;

a header storage area configured to store header portions of said one or more packets;

a first re-assembly storage area configured to store data portions of said one or more packets;

a second re-assembly storage area;

a code generator configured to generate a code for the first packet to indicate whether the first packet is re-assembleable with a second packet in the second communication flow; and

a transfer module configured to:

transfer a data portion of a first packet of a second communication flow into said second re-assembly storage area if said first packet is larger than a predetermined size; and

otherwise, transfer said first packet into said header storage area.

Count #34 (corresponds to Claim 36 of the '489 patent):

A network interface for transferring a packet from a network to a host computer, comprising:

a parser configured to examine one or more packets received from a network;

a flow manager configured to manage a first communication flow comprising said one or more packets;

a header storage area configured to store header portions of said one or more packets;

a first re-assembly storage area configured to store data portions of said one or more packets;

a second re-assembly storage area;

a code generator configured to generate a code for the first packet to indicate whether the first packet is larger than the predetermined size; and

a transfer module configured to:

transfer a data portion of a first packet of a second communication flow into said second re-assembly storage area if said first packet is larger than a predetermined size; and

otherwise, transfer said first packet into said header storage area.

Count #35 (corresponds exactly to Claim 37 of the '489 patent):

A communication interface configured to store packets for transfer to a host computer, comprising:

a parser configured to determine whether a packet includes a data portion;

a re-assembly storage area configured to store data portions of a plurality of packets from a single communication flow; and

a hybrid storage area configured to store:

header portions of the plurality of packets; and

one or more packets smaller than a first predetermined size.

Count #36 (corresponds to Claim 39 of the '489 patent):

A communication interface configured to store packets for transfer to a host computer, comprising:

a parser configured to determine whether a packet includes a data portion;

a re-assembly storage area configured to store data portions of a plurality of packets from a single communication flow;

a hybrid storage area configured to store:

header portions of the plurality of packets; and

one or more packets smaller than a first predetermined size; and

a code generator configured to generate a code for each packet received at the communication interface;

wherein said code is configured to indicate a type of storage area in which said packet may be stored.

Count #37 (corresponds to Claim 40 of the '489 patent):

A communication interface configured to store packets for transfer to a host computer, comprising:

a parser configured to determine whether a packet includes a data portion;

a re-assembly storage area configured to store data portions of a plurality of packets from a single communication flow; and

a hybrid storage area configured to store:

header portions of the plurality of packets; and

one or more packets smaller than a first predetermined size;

wherein said parser is further configured to determine a size of each packet received at the communication interface.

Per rule 607(a)(3), the claims in the patent (6,480,489) correspond to the counts as set forth in the table below. Per rule 607(a)(4), the claims in the present application correspond exactly to the counts as set forth in the table below:

Count No.	Corresponding Claim in USP 6,427,169	Corresponding Claim in the present application
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	18	18
18	19	19
19	20	20
20	21	21
21	22	22
22	23	23

23	24	24
24	25	25
25	26	26
26	27	27
27	28	28
28	29	29
29	30	30
30	31	31
31	32	32
32	33	33
33	35	35
34	36	36
35	38	38
36	39	39
37	40	40

Per rule 607(a)(5), application of the terms of Claims 1-16, 18-33 and 25-40 to the disclosure of the above-identified application is set forth in the claims chart below.

USP 6,480,489 Claim Element	Support in Alacritech Application Serial No. 60/098,296 (Page #: Line #)
1. A method of storing a portion of a packet in a host computer memory, comprising:	
receiving a first packet at a communication interface;	(6:30-31) "When a ***frame*** is ***received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not."
receiving a second packet at said communication interface;	(38:17) "4.7.2.2. Two types of ***receive packets***"

storing a header portion of said first packet in a hybrid storage area of a host computer;	(13:17-18) "As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host." (13:4-5) "If a received ***frame*** fits in a ***small buffer***, the INIC will use a small buffer. Otherwise it will use a large buffer."
if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and	(5:27-28) "In the fast path case, network ***data*** is given to the ***host*** after the headers have been processed and stripped." (13:17-21) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host. The host uses the header to determine what further data is following, ***allocates the necessary host buffers, and these are passed back to the INIC*** via a command to the INIC. ***The INIC then fills these buffers from data it was accumulating*** on the card..." (13:22-24) "This ... puts all the data into the header buffer or, if the header buffer is too small, uses a ***large (2K) host buffer for all the data.***"
if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area.	(12:44 to 13:4-5) "In order to ***receive 1514 byte frames*** (maximum ether frame size), however, we can only ***fit 2 buffers in a 4k page***, which is not a substantial savings. Fortunately, network frames tend to be either large (***~1500 bytes***), or small (***<256 bytes***). We take advantage of this fact by allocating large and small receive buffers. ***If a received frame fits in a small buffer, the INIC will use a small buffer. Otherwise it will use a large buffer.***" (21: 20-22) "For ***segments less than a full 1460 byte payload***, all of the received segment will be forwarded; it will be absorbed directly by the TDI client without any further MDL exchange." (7:28-38) "Clearly there are circumstances in which this does not make sense. When a ***small amount of data*** (500 bytes for example), with a push flag set indicating that the data must be delivered to the client immediately, it does not make sense to deliver some of the data directly while waiting for the list of addresses to DMA the rest. Under these circumstances, it makes more sense to deliver the 500 bytes directly to the host, and allow the host to copy it into its final destination. While various ranges are feasible, it is currently preferred that anything ***less than a segment's (1500 bytes) worth of data*** will be ***delivered directly to the host***, while anything more will be delivered as a small piece (which may be 128 bytes), while waiting until receiving the destination memory address before moving the rest."
2. The method of claim 1, wherein said hybrid storage area is substantially equal in size to one memory page of said host computer.	(12:42-44) "In the driver we allocate ***a block of contiguous memory (typically a page***, which is typically 4k)." (14:27) "***Receive data buffers are allocated in blocks of 2, 2k bytes each (4k page).***"

3. The method of claim 1, wherein said predetermined size is approximately 256 bytes.	(14:6-7) "***Header buffers*** in host memory are ***256 bytes long***, and are aligned on ***256 byte*** boundaries."
4. The method of claim 1, further comprising:	
receiving a code associated with said first packet, said code indicating that a data portion of said first packet is re-assembleable;	(6:36-38) "The ***header is fully parsed*** by hardware and its ***type is summarized*** in a single status word. (41:25-27) "2. If the ***type field contains our custom INIC type*** (TCP for example): A. If the ***header buffer specifies a fast-path connection***, allocate one or more mbufs headers to map the header and possibly data buffers."
wherein said data storage area is a re-assembly data storage area.	(13:17-21) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host. The host uses the header to determine what further data is following, ***allocates the necessary host buffers***, and these are passed back to the INIC via a command to the INIC. The ***INIC then fills these buffers from data*** it was accumulating on the card and notifies the host by sending a response to the command."
5. The method of claim 1, further comprising:	
receiving a code associated with said second packet, said code indicating that a data portion of said second packet is not re-assembleable;	(85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer contains a bit (bit 29) that informs the microcode if this frame is definitely not a fast-path candidate...***" (85:38) "If bit 29 is set, this frame is going ***slow-path***."
wherein said data storage area is a non-re-assembly data storage area.	(87:9-15) "The following is a summary of ***slow-path processing***: • Examine frame status bytes to determine if frame is in-error; if so, only these status bytes will be sent to the host. • ***Move the frame into either a small or a large host buffer*** via DMA. It is not split across these buffers. • Set frame status and address details and DMA status to the host. • Send event to the Utility processor to post Receive status in the ISR."
6. The method of claim 1, further comprising receiving a code with said first packet, said code indicating that said first packet does not contain a data portion.	(14:6-7) "There will be a ***field*** in the ***header buffer*** indicating it has valid ***data***." (86:23-29) "Examine the ***frame header*** to generate an event from it. The Receive events that can be generated on a given context from a frame are: ***receive a pure ACK...*** ***receive a window probe...***"

7. The method of claim 1, further comprising receiving a code with said second packet, said code indicating that said second packet is smaller than said predetermined size.	(70:12-13) "***Receive data*** is passed from the INIC to the host by filling in a header buffer. The ***header buffer contains information*** about the data, such as the ***length.***"
8. The method of claim 1, further comprising padding said hybrid storage area to align said header portion with a predetermined index in said hybrid storage area.	(14:6-7) "***Header buffers*** in host memory are 256 bytes long, and are ***aligned on 256 byte boundaries.***"
9. The method of claim 1, wherein said first packet belongs to a first communication flow and said second packet belongs to a second communication flow.	(6:10-16) "This introduces the notion of a ***Communication Control Block (CCB)*** cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time."
10. A method of transferring multiple packets in a communication flow into a host computer, comprising: receiving a first packet at a network interface for transfer to a host computer; identifying a communication flow comprising said first packet;	 (6:30-31) "When a ***frame*** is ***received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not." (86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports.*** This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to ***determine if a CCB exists on the INIC for this frame.*** It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***" (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection.*** This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.***

	(4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports.***"
receiving a second packet at said network interface, wherein said communication flow further comprises said second packet;	See above.
associating a code with said first packet to indicate whether a data portion of said first packet is re-assembleable with a data portion of another packet in said communication flow;	(85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue. The pointer contains ***a bit (bit 29) that informs*** the microcode if this frame is definitely ***not a fast-path candidate*** (e.g., not TCPIP, or has an error of some sort)." (86:1-9) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already ***generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports.*** This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The ***microcode uses the hash to determine if a CCB exists on the INIC for this frame.*** It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then ***the frame will be processed against the CCB*** by the INIC. If not, then the frame is sent for slow-path processing."
receiving a set of descriptors from the host computer, wherein an aggregate size of said set of descriptors approximates a cache line size of the host computer;	(145:5-1) "The read multiple command requires that the Pmi sequencer be capable of transferring a ***cache line*** or more of data. To accomplish this end, Pmi will automatically perform partial cache line bursts until it has ***aligned the transfers on a cache line boundary*** at which time it will begin usage of the read multiple command. The Sram fifo depth, of ***256 bytes***, has been chosen in order to allow Pmi to accommodate ***cache line sizes up to 128 bytes***. Provided the cache line size is less than 128 bytes, Pmi will perform ***multiple, contiguous cache line bursts*** until it has filled the fifo." (12:41-45) "Thus we needed an efficient way in which to pass receive buffer addresses to the INIC. We accomplished this by ***passing a block of receive buffers to the INIC*** at one time. In the driver we allocate a block of contiguous memory (typically a page, which is typically 4k). We ***write the address of that block to the INIC with the bottom bits of the address specifying the number of buffers*** in the block." (14:6-13) "Header buffers in host memory are 256 bytes long, and are aligned on 256 byte boundaries. There will be a field in the header buffer indicating it has valid data. This field will initially be reset by the host before passing the buffer descriptor to the INIC. A ***set of header***

	buffers are passed from the host to the INIC by the host writing to the Header Buffer Address Register on the INIC. This register is defined as follows: ***Bits 31-8*** Physical address in host memory of the first of a set of contiguous header buffers ***Bits 7-0*** Number of header buffers passed."
storing a header portion of said first packet in a header storage area;	(13:17-18) "As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host."
storing said data portion of said first packet in a re-assembly storage area; and	(13:18-21) "The host uses the header to determine what further data is following, ***allocates the necessary host buffers***, and these are passed back to the INIC via a command to the INIC. The INIC then ***fills these buffers from data it was accumulating*** on the card and notifies the host by sending a response to the command." (10:25-32) "the NetBIOS client will ***allocate a memory pool*** large enough to hold the entire NetBIOS message, and will pass this address or set of addresses down to the transport driver. The transport driver will allocate an INIC command buffer, fill it in with the list of addresses, set the command type to tell the INIC that this is where to put the receive data, and then pass the command off to the INIC by writing to the command register. When the INIC receives the command buffer, it will ***DMA*** the remainder of the NetBIOS ***data***, as it is received, ***into the memory address or addresses designated by the host***."
storing a data portion of said second packet in said re-assembly storage area.	See above.
11. The method of claim 10, wherein said re-assembly storage area is substantially equal in size to one memory page of said host computer.	(9:23-24) "In order to avoid having to write to the INIC for every receive frame, we instead allow the host to pass off a ***pages worth (4k)*** of buffers in a single write." (9:30-37) "Large buffers are 2k in size. They are used to contain any fast or slow-path data that does not fit in a small buffer. Note that when we have a large fast-path receive, a small buffer will be used to indicate a small piece of the data, while the remainder of the data will be ***DMA'd directly into memory***. Large buffers are never passed to the host by themselves, instead they are always accompanied by a small buffer which contains status about the receive along with the large buffer address. By operating in the manner, the driver must only maintain and process the small buffer queue. Large buffers are returned to the host by virtue of being attached to small buffers. Since large buffers are 2k in size they are passed to the INIC 2 buffers at a time."

12. The method of claim 10, wherein said re-assembly storage area is used to store only data portions of packets in said communication flow.	(5:27-28) "In the fast path case, network ***data is given to the host after the headers have been processed and stripped.***"
13. The method of claim 10, further comprising retrieving an identifier of a first storage area in a host computer memory.	(7:23-27) "We will make use of this feature by providing a small amount of any received data to the host, with a notification that we have more data pending. When this small amount of data is passed up to the client, and it ***returns with the address in which to put the remainder of the data***, our host transport driver will pass that address to the INIC which will DMA the remainder of the data into its final destination." (21:33-41) "In the "large data input" case, where "bytes available" exceeds the packet length, the TDI client then provides an MDL, associated with an IRP, which must be completed when this MDL is filled. (This IRP/MDL may come back either in the response to ATCP's call of the receive handler, or as an explicit TDI_RECEIVE request.) The ATCP driver builds a "receive request" from the MDL information, and passes this to the INIC. This request contains: The TCP context identifier. Size and offset information. A ***scatter/gather list of physical addresses corresponding to the MDL pages.***"
14. The method of claim 13, further comprising: placing said first storage area identifier in a data structure configured to hold storage area identifiers; wherein said first storage area identifier is identifiable by an index in said data structure.	 (14:27-36) "Receive data buffers are allocated in blocks of 2, 2k bytes each (4k page). In order to pass receive data buffers to the INIC, the host must write two values to the INIC. The first value to be written is the ***Data Buffer Handle.*** The buffer handle is not significant to the INIC, but will be copied back to the host to return the buffer to the host. The second value written is the ***Data Buffer Address***. This is the physical address of the data buffer. When both values have been written, the INIC will ***add these values to FreeType queue of data buffer descriptors***. The INIC will extract 2 entries each time when dequeuing. Data buffers will be allocated and used by the INIC as needed. For each data buffer used, the data buffer handle will be copied into a header buffer. Then the header buffer will be returned to the host."
15. The method of claim 14, further comprising using said index to identify said first storage area for storing a portion of said first packet.	(14:34-36) "Data buffers will be allocated and used by the INIC as needed. For each data buffer used, the ***data buffer handle will be copied into a header buffer***. Then the header buffer will be returned to the host."
16. The method of claim 14, wherein said first storage area comprises one of said header storage area and said re-	

assembly storage area, the method further comprising: configuring a descriptor to store said index of said first storage area identifier to inform said host computer of the use of said first storage area to store one of said header portion and said data portion.	(14:15-17) "For each interface, the INIC will maintain a queue of these header ***descriptors*** in the SmallHType queue in it's own local memory, adding to the end of the queue every time the host writes to one of the Header Buffer Address Registers."
18. The method of claim 9, further comprising parsing a header portion of said first packet.	(6:36-37) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word."
19. The method of claim 18, wherein said parsing comprises: retrieving an identifier of a source of said first packet; and retrieving an identifier of a destination of said first packet.	(6:36-38) "The header is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the ***IP addresses and TCP ports*** to expedite CCB lookup." (160:6-7) "The ***Mac, network, transport and session information is analyzed*** as each byte is received and stored in the assembly register (AssyReg)." (86:1-3) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports.***
20. The method of claim 19, wherein said identifying a communication flow comprises assembling said source identifier and said destination identifier to form a flow key.	(6:10-14) "CCBs are ***initialized*** by the host during ***TCP connection setup.***" (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection.*** This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.*** (4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports.***" (86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports.*** This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
21. The method of claim 20, said identifying further comprising obtaining an index of said flow key in a database of flow keys.	(86:1-8) "The receive sequencer has already generated a ***hash*** based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports.*** This ***hash*** is used to ***index*** directly into a hash table on the INIC that ***points to entries in a CCB header table.*** The header table entries are chained on the hash table

	entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame." (24:36-39) "The initial command from ATCP to INIC expresses an "intention" to hand out the context. It carries a ***context number***, context numbers are allocated by the ATCP driver, which keeps a per-INIC ***table of free and in-use context numbers***. It also includes the ***source and destination IP addresses and ports***, which will allow the INIC to establish a "provisional" context."
22. The method of claim 9, wherein said identifying comprises: receiving a virtual connection identifier, wherein a communication flow comprising said first packet can be identified by said virtual connection identifier.	 24:36-39) "The initial command from ATCP to INIC expresses an "intention" to hand out the context. It carries a ***context number***, context numbers are allocated by the ATCP driver, which keeps a per-INIC ***table of free and in-use context numbers***. It also includes the ***source and destination IP addresses and ports***..."
23. The method of claim 9, wherein said storing a header portion comprises: retrieving an index of a header storage area of a host computer; retrieving an identifier of a location in said header storage area; and storing a header portion of said first packet at said location in said header storage area.	 (14:9-11) "A set of header buffers are passed from the host to the INIC by the host writing to the ***Header Buffer Address Register*** on the INIC. (14:11-14) "This register is defined as follows: Bits 31-8 ***Physical address in host memory of the first of a set of contiguous header buffers*** Bits 7-0 Number of header buffers passed." See above.
24. The method of claim 23, wherein said header storage area index comprises an index of said header storage area within a collection of storage area identifiers.	(14:28-30) "In order to pass receive data buffers to the INIC, the host must write two values to the INIC. The first value to be written is the Data Buffer Handle. The buffer handle is not significant to the INIC, but will be copied back to the host to return the buffer to the host. The second value written is the Data Buffer Address. This is the physical address of the data buffer. When both values have been written, the INIC will add these values to FreeType queue of data buffer descriptors. The INIC will extract 2 entries each time when dequeuing."
25. The method of claim 24, wherein said identifier of a location in said header storage	(14:12-14) "Bits 31-8 Physical address in host memory of the first of a set of contiguous header buffers."

area comprises an address within said header storage area.	
26. The method of claim 9, wherein said storing said data portion comprises: retrieving an identifier of a re-assembly storage area of a host computer; retrieving an identifier of a location in said re-assembly storage area; and storing said data portion of said first packet at said location in said re-assembly storage area.	 (7:24-26) "When this small amount of data is passed up to the client, and it ***returns with the address in which to put the remainder of the data***, our host transport driver will pass that address to the INIC..." (10:25-32) "Meanwhile, the NetBIOS client will ***allocate a memory pool*** large enough to hold the entire NetBIOS message, and will pass this ***address or set of addresses*** down to the transport driver. The transport driver will allocate an INIC command buffer, fill it in with the ***list of addresses***, set the command type to tell the INIC that this is where to put the receive data, and then pass the command off to the INIC by writing to the command register. (7:24-27) "When this small amount of data is passed up to the client, and it returns with the address in which to put the remainder of the data, our host transport driver will pass that address to the INIC which will ***DMA the remainder of the data*** into its ***final destination***."
27. The method of claim 26, wherein said retrieving an identifier of a re-assembly storage area comprises reading an entry in a re-assembly storage table, said entry corresponding to said communication flow.	(21:37-39) "The ATCP driver builds a "receive request" from the MDL information, and passes this to the INIC. This request contains: The ***TCP context identifier.*** Size and offset information. A scatter/gather list of physical addresses corresponding to the MDL pages."
28. The method of claim 18, wherein said data portion is of unknown size prior to said parsing.	(162:6-10) "27:00 address Represents the last address +1 to which frame data was transferred. This ***can be used to determine the size of the frame received*** by first subtracting one from the address then comparing with the buffer boundary as indicated by the size bits." (165:20) "14:00 LengthCnt Refer to E110 Technical Manual."
29. A network interface for transferring a packet from a network to a host computer, comprising: a parser configured to examine one or more packets received from a network;	 (6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup."

a flow manager configured to manage a first communication flow comprising said one or more packets;	(85:27-41) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue. The pointer contains a bit (bit 29) that informs the microcode if this frame is definitely not a fast-path candidate (e.g., not TCPIP, or has an error of some sort). Receive frame processing involves extracting this pointer from the Receive hardware queue, and setting up a DMA into an SRAM header buffer of the first X bytes from the DRAM frame buffer. The size of the DMA is determined by whether bit 29 is set or not. If it is set (this frame is not a fast-path candidate), then only the status bytes are needed by the microcode, so the size would be 16 bytes. Otherwise up to 92 bytes are DMA'd – sufficient to get all useful headers. When this DMA is complete, the status bytes are used by the microcode to determine whether to jump to fast-path or slow-path processing."
a header storage area configured to store header portions of said one or more packets;	(13:17-18) "As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host."
a first re-assembly storage area configured to store data portions of said one or more packets;	(5:27-28) "In the fast path case, network ***data*** is given to the host after the headers have been processed and stripped." (13:17-21) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host. The host uses the header to determine what further data is following, ***allocates the necessary host buffers, and these are passed back to the INIC via a command to the INIC. The INIC then fills these buffers from data it was accumulating*** on the card..."
a second re-assembly storage area; and	See above.
a transfer module configured to:	
transfer a data portion of a first packet of a second communication flow into said second re-assembly storage area if said first packet is larger than a predetermined size; and	(13:20-21) "Alternatively, the fast-path may receive a ***header and data*** that is a complete request, but that is also too large for a header buffer. This results in a ***header*** and ***data buffer*** being passed to the host." (14:1-3) "Note that the order in which data is written is important. ***Data buffers are moved by DMA*** into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived."
otherwise, transfer said first packet into said header storage area.	(7:28-38) "Clearly there are circumstances in which this does not make sense. When a ***small amount of data*** (500 bytes for example), with a push flag set indicating that the data must be delivered to the client

	immediately, it does not make sense to deliver some of the data directly while waiting for the list of addresses to DMA the rest. Under these circumstances, it makes more sense to deliver the 500 bytes directly to the host, and allow the host to copy it into its final destination. While various ranges are feasible, it is currently preferred that anything ***less than a segment's (1500 bytes) worth of data*** will be ***delivered directly to the host***, while anything more will be delivered as a small piece (which may be 128 bytes), while waiting until receiving the destination memory address before moving the rest."
30. A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet from a network to a host computer, the method comprising:	(110:12) "The ***processor instructions reside in the on-chip control-store, which is implemented as a mixture of ROM and Sram.***" (80:6-31) "As specified in other sections, the INIC supplies a set of 3 custom processors (CPUs) that provide considerable hardware-assist to the microcode running thereon... • Multiple register contexts or process slots with register access controlled by simply setting a process register. The Protocol Processor will provide 512 SRAM-based registers to be shared among the 3 CPUs in any way desired. The current implementation uses 16 processes of 16 registers each, leaving 256 scratch registers to be shared. • A set of CPU-specific registers that are the same local-cpu register number, but for which the real register is determined by an offset based on the CPU number; this allows multiple CPUs to execute the same code at the same time without register clashes or interlocks. These registers are a part of the above-mentioned scratch pool."
receiving a first packet at a communication interface;	((6:30-31) "When a ***frame*** is ***received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not."
receiving a second packet at said communication interface;	(38:17) "4.7.2.2. Two types of ***receive packets***"
storing a header portion of said first packet in a hybrid storage area of a host computer;	(13:17-18) "As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host." (13:4-5) "If a received ***frame*** fits in a ***small buffer***, the INIC will use a small buffer. Otherwise it will use a large buffer."
if said first packet includes a data portion, storing said data portion in a data storage area of said host computer; and	(5:27-28) "In the fast path case, network ***data*** is given to the ***host*** after the headers have been processed and stripped." (13:17-21) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host. The host uses the header to determine what further data is following, ***allocates the necessary host buffers, and these are passed back to the INIC*** via a command to the INIC. ***The INIC then fills these buffers from data it was accumulating*** on the card..."

	(13:22-24) "This ... puts all the data into the header buffer or, if the header buffer is too small, uses a ***large (2K) host buffer for all the data.***"
if said second packet is smaller than a predetermined size, storing said second packet in said hybrid storage area.	(12:44 to 13:4-5) "In order to ***receive 1514 byte frames*** (maximum ether frame size), however, we can only ***fit 2 buffers in a 4k page***, which is not a substantial savings. Fortunately, network frames tend to be either large (***~1500 bytes***), or small (***<256 bytes***). We take advantage of this fact by allocating large and small receive buffers. ***If a received frame fits in a small buffer, the INIC will use a small buffer. Otherwise it will use a large buffer.***" (21: 20-22) "For ***segments less than a full 1460 byte payload***, all of the received segment will be forwarded; it will be absorbed directly by the TDI client without any further MDL exchange." (7:28-38) "Clearly there are circumstances in which this does not make sense. When a ***small amount of data*** (500 bytes for example), with a push flag set indicating that the data must be delivered to the client immediately, it does not make sense to deliver some of the data directly while waiting for the list of addresses to DMA the rest. Under these circumstances, it makes more sense to deliver the 500 bytes directly to the host, and allow the host to copy it into its final destination. While various ranges are feasible, it is currently preferred that anything ***less than a segment's (1500 bytes) worth of data*** will be ***delivered directly to the host***, while anything more will be delivered as a small piece (which may be 128 bytes), while waiting until receiving the destination memory address before moving the rest."
31. A method of transferring a packet from a communication link to a host computer, comprising:	
parsing a first packet received from a communication link to determine if the first packet conforms to a predetermined set of communication protocols;	(6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup." (6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc)."
generating, from identifiers of a source and a destination of the first packet extracted from the first packet, a flow key to identify a communication flow comprising the first packet;	(6:10-14) "CCBs are ***initialized*** by the host during ***TCP connection setup.***" (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.***

	(4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports.***"
updating a flow database to track a status of the communication flow;	(86:7-8) "If a ***match is found***, then the frame will be ***processed against the CCB*** by the INIC." (6:10-16) "A ***CCB*** is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the ***current send and receive sequence numbers...***"
associating a first operation code with the first packet to indicate whether the first packet is re-assembleable with another packet in the communication flow;	(85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status.*** A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:38) "***If bit 29 is set, this frame is going slow-path.***"
maintaining a plurality of storage areas for transferring packets to a host computer, including:	(9:22-30) "We will instead write receive buffer addresses to the INIC as receive buffers are filled. In order to avoid having to write to the INIC for every receive frame, we instead allow the host to pass off a pages worth (4k) of buffers in a single write. 2.3.2. ***Support small and large buffers on receive*** In order to reduce further the number of writes to the INIC, and to reduce the amount of memory being used by the host, we support two different buffer sizes. A small buffer contains roughly 200 bytes of data payload, as well as extra fields containing status about the received data bringing the total size to 256 bytes. We can therefore pass 16 of these small buffers at a time to the INIC. Large buffers are 2k in size."
a first storage area configured to store data portions of packets in the communication flow;	(9:31-33) "Note that when we have a large fast-path receive, a small buffer will be used to indicate a small piece of the data, while the remainder of the ***data will be DMA'd directly into memory***. (10:25-32) "Meanwhile, the NetBIOS client will ***allocate a memory pool large enough to hold the entire NetBIOS message***, and will pass this address or set of addresses down to the transport driver. The transport driver will allocate an INIC command buffer, fill it in with the list of addresses, set the command type to tell the INIC that this is where to put the receive data, and then pass the command off to the INIC by writing to the command register. When the INIC receives the command buffer, it will DMA the remainder of the NetBIOS data, as it is received, into the memory address or addresses designated by the host."

a second storage area configured to store a packet having an associated operation code indicating that the packet is not re-assembleable; and a third storage area configured to store packets less than a predetermined size and headers of packets having data portions stored in said first storage area; and storing the first packet in one or more of said plurality of storage areas for transfer to the host computer.	(11:2-6) "If the INIC receives a frame that ***does not contain a TCP segment*** for one of its CCB's, it simply passes it to the host as if it were a dumb NIC. If the frame fits into a small buffer (~200 bytes or less), then it simply fills in the small buffer with the data and notifies the host. Otherwise it places the data in a large buffer, writes the address of the large buffer into a small buffer, and again notifies the host." (13:4-5) "If a received frame ***fits in a small buffer***, the INIC will use a small buffer. Otherwise it will use a large buffer." (13:17-18) "As mentioned above, the fast-path flow puts a ***header into a header buffer*** that is then forwarded to the host." See above.
32. The method of claim 31, wherein said storing comprises: selecting one or more storage areas of said plurality of storage areas on the basis of said operation code.	 (85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status.*** A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:38) *"If bit 29 is set, this frame is going slow-path."*
33. The method of claim 31, wherein each of said plurality of storage areas is substantially equal in size to one memory page of the host computer.	(9:22-37) "We will instead write receive buffer addresses to the INIC as receive buffers are filled. In order to avoid having to write to the INIC for every receive frame, we instead allow the host to pass off ***a pages worth (4k) of buffers*** in a single write. 2.3.2. Support small and large buffers on receive In order to reduce further the number of writes to the INIC, and to reduce the amount of memory being used by the host, we support two different buffer sizes. A small buffer contains roughly 200 bytes of data payload, as well as extra fields containing status about the received data bringing the total size to ***256 bytes. We can therefore pass 16 of these small buffers at a time to the INIC. Large buffers are 2k in size***...***Since large buffers are 2k in size they are passed to the INIC 2 buffers at a time.***"
35. The network interface of claim 29, further comprising: a code generator configured to generate a code for the first	 (85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the ***Receive hardware sequencer***, along with 16 bytes of the

packet to indicate whether the first packet is re-assembleable with a second packet in the second communication flow.	above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:38) ***"If bit 29 is set, this frame is going slow-path."***
36. The network interface of claim 29, further comprising: a code generator configured to generate a code for the first packet to indicate whether the first packet is larger than the predetermined size.	 (162:6-10) "27:00 address Represents the last address +1 to which frame data was transferred. This can be used to determine the size of the frame received by first subtracting one from the address then comparing with the buffer boundary as indicated by the size bits." (165:20) "14:00 LengthCnt Refer to E110 Technical Manual."
37. A communication interface configured to store packets for transfer to a host computer, comprising: a parser configured to determine whether a packet includes a data portion; a re-assembly storage area configured to store data portions of a plurality of packets from a single communication flow; and a hybrid storage area configured to store: header portions of the plurality of packets; and	 (6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup." (6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc)." (9:31-33) "Note that when we have a ***large fast-path receive***, a small buffer will be used to indicate a small piece of the data, while the remainder of the ***data will be DMA'd directly into memory***. (10:25-32) "Meanwhile, the NetBIOS client will allocate a memory pool large enough to hold the entire NetBIOS message, and will pass this address or set of addresses down to the transport driver. The transport driver will allocate an INIC command buffer, fill it in with the list of addresses, set the command type to tell the INIC that this is where to put the receive data, and then pass the command off to the INIC by writing to the command register. When the INIC receives the command buffer, it will DMA the remainder of the NetBIOS data, as it is received, into the memory address or addresses designated by the host." (13:17-18) "As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host."

one or more packets smaller than a first predetermined size.	(13:4-5) "If a ***received frame fits in a small buffer***, the INIC will use a small buffer. Otherwise it will use a large buffer."
39. The communication interface of claim 37, further comprising: a code generator configured to generate a code for each packet received at the communication interface; wherein said code is configured to indicate a type of storage area in which said packet may be stored.	 (85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:38) ***"If bit 29 is set, this frame is going slow-path."***
40. The communication interface of claim 37, wherein said parser is further configured to determine a size of each packet received at the communication interface.	(162:6-10) "27:00 address Represents the last address +1 to which frame data was transferred. This can be used to determine the size of the frame received by first subtracting one from the address then comparing with the buffer boundary as indicated by the size bits." (165:20) "14:00 LengthCnt Refer to E110 Technical Manual."

Claims 17, 34 and 38:

Replicated below are copies of the claims in the present application that are not copied from U.S. Patent No. 6,480,489 (Claims 17, 34 and 38).

17. A method of network communication, the method comprising:

providing a computer having a processor and a memory, the memory including first, second and third storage areas that are accessible by a communication interface;

receiving, by the communication interface, a first packet, and storing the first packet in the first storage area;

receiving, by the communication interface, a second packet, storing a header of the second packet in the first storage area, and storing data of the second packet in the second storage area; and

receiving, by the communication interface, a third packet, and storing data of the third packet in the third storage area, the third storage area containing data from a plurality of packets and corresponding to an application running on the computer, the third storage area containing no headers.

34. The method of claim 17, further comprising associating the third storage area with a transmission control protocol (TCP) connection.

38. The method of claim 17, further comprising sending a command from the computer to the communication interface, the command including an address of the third storage area.

CONCLUSION

In view of the above remarks, Applicants request that an interference be declared between the present application and U.S. Patent No. 6,480,489. If the Examiner would like to discuss any aspect of this application, including how the claims are supported by Applicants' specification, the Examiner is requested to call the undersigned at (925) 484-9295.

Respectfully submitted,

Mark Lauer
Reg. No. 36,578
6601 Koll Center Plaza,
Suite 245
Pleasanton, CA 94566
Tel: (925) 484-9295
Fax: (925) 484-9291

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail, Label No. EK916850044US, in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450, on August 4, 2003.

Date: 8-4-03

Mark Lauer